Inventor: Short
Docket No.: 11986/59946

METHOD AND APPARATUS FOR REMOTE DIGITAL KEY GENERATION

This Application is a Continuation-In-Part of currently pending Short, et al. – Method and Apparatus for Secure Digital Chaotic Communication – Application Serial No. 09/436,910 filed November 9, 1999.

FIELD OF INVENTION

The present invention relates generally to a method and apparatus for the remote generation of a digital key to an encryption system. More specifically, it relates to a system for the remote generation of such a digital key through the use of chaotic systems.

BACKGROUND OF INVENTION

In many secure communication systems known to those skilled in the art, a key, or series of keys, are processed according to a given algorithm with a plain text to produce an encrypted text. It is assumed that the algorithm is well known. Thus, the security of the encrypted text is dependent on the security of the key.

It follows that the secure transmission of the key, from the party encrypting the plain text to the party decrypting the encrypted text, is of great importance. In many cases, the key is sent by means other than those used to send the encrypted text. Alternatively, the key itself is not transmitted, but some signal is transmitted by the

encrypting party that allows for the remote generation of the key by the decrypting party. Chaotic systems can be used for the remote generation of a digital key.

In general, a chaotic system is a dynamical system which has no periodicity and the final state of which depends so sensitively on the system's precise initial state that its time-dependent path is, in effect, long-term unpredictable even though it is deterministic. Identical chaotic systems can be distributed in a secure manner to an encrypting party and a decrypting party. When the encrypting party desires to communicate a digital key to the decrypting party, a number of different systems can be used to cause the digital key to be generated remotely by the decrypting party without transmitting the digital key itself.

In one method, described in Short, *et al* –Method and Apparatus for Secure Digital Chaotic Communication – Application No. 09/436,910 filed November 9, 1999 and incorporated herein by reference ("Short *et al.*–Chaotic Communication"), a bitstream is selected by the encrypting party for use as a digital key and is then generated remotely by the decrypting party. Controls are intermittently applied by a transmitter-encoder to a chaotic system to generate a bitstream corresponding to the digital key. A control/no control bitstream is thereby created in which a 0 indicates that no control was applied and a 1 indicates that a control was applied. The control/ no control bitstream and a prepended synchronization bitstream are transmitted to a receiver-decoder. An identical chaotic system in the receiver-decoder is driven into synchrony and is then subject to intermittent controls based on the control/ no control bitstream, thereby causing the identical chaotic system to generate the digital key.

The method of the present invention does not start with the selection of a bitstream for use as a ditigal key. Instead, a chaotic system in an encryptor is allowed to

generate an unpredictable bitstream, for use as a digital key, which bitstream is then also generated remotely at an identical chaotic system in a decryptor. An initialization code is sent by an encryptor to a chaotic system, which is then allowed to generate an unpredictable key bitstream. The same initialization code is sent to an identical chaotic system in a decryptor to drive that chaotic system into synchrony. The synchronized chaotic system is then allowed to generate a key bitstream, which is identical to the other key bitstream because the chaotic systems have been synchronized. The initialization code, if it is intercepted, cannot be used to reproduce either the key bitstream or the chaotic system.

SUMMARY OF THE INVENTION

An object of the present invention is to provide a new method and apparatus for remote digital key generation. Another object of the present invention is to provide a faster, more secure method and apparatus for remote digital key generation through the use of chaotic systems.

The present invention may be implemented either in hardware or software. An initialization code is sent to a first chaotic system that is then allowed to generate an unpredictable sequence of bits 0 and 1 for use as a digital key. The same initialization code is then sent to a second chaotic system, identical to the first chaotic system, to drive the second chaotic system into synchrony. The second chaotic system is then allowed to generate a sequence of bits 0 and 1, which is identical to the first sequence of bits because the two chaotic systems have been synchronized. Thus, a bitstream, suitable for use as a

digital key, has been generated remotely, without the transmission of the digital key or any information from which the digital key or the chaotic system can be reconstructed.

The foregoing and other objects, features and advantages of the present invention will be apparent from the following more detailed description of preferred embodiments of the invention as illustrated in the accompanying drawings.

IN THE DRAWINGS

Fig. 1 is a block diagram of a remote digital key generation system according to an embodiment of the present invention.

Fig. 2 is a flow chart showing the remote generation procedures of the remote digital key generation system shown in Fig. 1.

Fig. 3 is a plot of the double scroll oscillator resulting from the given differential equations and parameters.

Fig. 4 is a plot of the symbolic dynamics function, r(x).

Fig. 5 is a plot of the Poincare Map for the given double scroll oscillator.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is based on two important attributes of chaotic systems. The first such attribute is that the trajectory of a chaotic system will visit different regions of the system over time. If the different regions of the system are labeled 0 or 1, an unpredictable bitstream will be generated by the trajectory, as is described in more detail below. Alternatively, the different regions of the system may be labeled with any symbols, and the trajectory will generate an unpredictable string of such symbols. The

string of symbols can then be converted to a bitstream according to an appropriate algorithm. For example, a string of letters and base 10 numbers can be converted using the ASCII algorithm. Controls can also be imposed on a chaotic system to cause it to generate a specific bitstream. The second such attribute is that certain controls may be used as initialization codes, as is also described in more detail below, to synchronize identical chaotic systems. The synchronized chaotic systems will then generate identical bitstreams.

In a preferred embodiment, such a chaotic system is a double-scroll oscillator [S. Hayes, C. Grebogi, and E. Ott, Communicating with Chaos, Phys, Rev. Lett. 70, 3031 (1993)], described by the differential equations

$$C_1\dot{v}_{C1} = G(v_{C2} - v_{C1}) - g(v_{C1})$$

$$C_2\dot{v}_{C2} = G(v_{C1} - v_{C2}) + i_L$$

$$L\dot{i}_L = -v_{C2},$$

where

$$g(v) = \begin{cases} m_1 v, & \text{if } -B_p \le v \le B_p; \\ m_0(v + B_p) - m_1 B_p, & \text{if } v \le -B_p; \\ m_0(v - B_p) + m_1 B_p, & \text{if } v \ge B_p \end{cases}$$

The attractor that results from a numerical simulation using the parameters $C_1 = 1/9$, $C_2 = 1$, $L = 1/7$, $G = 0.7$, $m_0 = -0.5$, $m_1 = -0.8$, and $B_p = 1$ has two lobes, which can be labeled 0 and 1, and each of which surrounds an unstable fixed point, as shown in Fig. 3.

Because of the chaotic nature of this oscillator's dynamics, it is possible to take advantage of sensitive dependence on initial conditions by carefully choosing small perturbations to direct trajectories around each of the loops of the attractor. In this way, a desired bit stream can be generated by steering the trajectories around the appropriate lobes of the attractor, suitably labeled 0 and 1. It should be noted that other embodiments could have more than two lobes, in which each lobe is labeled 0 or 1 or a symbol from any chosen symbol set.

There are a number of means to control the chaotic oscillator in this preferred embodiment to specify the bits 0 and 1 more precisely. In a further preferred embodiment, a Poincare surface of section is defined on each lobe by intersecting the attractor with the half planes $i_L = \pm GF, |v_{C1}| \leq F$, where $F = B_p(m_0 - m_1)/(G+m_0)$. When a trajectory intersects one of these sections, the corresponding bit can be recorded. Then, a function r(x) is defined, which takes any point on either section and returns the future symbolic sequence for trajectories passing through that point. If l_1, l_2 , l_3, ... represent the lobes that are visited on the attractor (so l_i is either a 0 or a 1), and the future evolution of a given point x_0 is such that $x_0 \rightarrow l_1, l_2, l_3, \ldots, l_N$ for some number N of loops around the attractor, then the function r(x) is chosen to map x_0 to an associated binary fraction, so $r(x_0) = 0.l_1 l_2 l_3 \ldots l_N$, where this represents a binary decimal (base 2). Then, when r(x) is calculated for every point on the cross-section, the future evolution of any point on the cross-section is known for N iterations. The resulting function is shown in Fig. 4, where r(x) has been calculated for 12 loops around the attractor.

Control of the trajectory begins when it passes through one of the sections, say at x_0. The value of $r(x_0)$ yields the future symbolic sequence followed by the current trajectory for *N* loops. If generation of a desired bit stream requires a different symbol

in the Nth position of the sequence, r(x) can be searched for the nearest point on the section that will produce the desired symbolic sequence. The trajectory can be perturbed to this new point, and it continues to its next encounter with a surface. It should be noted that this embodiment exhibits a "limited grammar," which means that not all sequences of 0's and 1's can be directly encoded, because the chaotic oscillator always loops more than once around each lobe. Consequently, a sequence of bits such as 00100 is not in the grammar since it requires a single loop around the 1-lobe. A simple remedy is to repeat every bit in the code or append a 1- or 0-bit to each contiguous grouping of 1- or 0-bits, respectively. Other embodiments may have a different grammar, and examples exist where there are no restrictions on the sequence of 0's and 1's. For this system, the bitstream is read from the oscillation of coordinate i_L, so the bit stream is read from the peaks and valleys in i_L (there are small loops/minor peaks that occur as the trajectory is switching lobes of the attractor, but these are ignored.)

The calculation of r(x) in the preferred embodiment was done discretely by dividing up each of the cross-sections into 2001 partitions ("bins") and calculating the future evolution of the central point in the partition for up to 12 loops around the lobes. As an example, controls were applied so that effects of a perturbation to a trajectory would be evident after only 5 loops around the attractor. In addition to recording r(x), a matrix M was constructed that contains the coordinates for the central points in the bins, as well as instructions concerning the controls at these points. These instructions simply tell how far to perturb the system when it is necessary to apply a control. For example, at an intersection of the trajectory with a cross-section, if $r(x_0)$ indicates that the trajectory

will trace out the sequence 10001, and sequence 10000 is desired, then a search is made for the nearest bin to x_0 that will give this sequence, and this information is placed in M. (If the nearest bin is not unique, then there must be an agreement about which bin to take, for example, the bin farthest from the center of the loop.) Because the new starting point after a perturbation has a future evolution sequence that differs from the sequence followed by x_0 by at most the last bit, only two options need be considered at each intersection, control or no control.

The matrix M holds the information about which bin should hold the new starting point for the perturbed trajectory. In an analog hardware implementation of the preferred embodiment, the perturbations are applied using voltage changes or current surges; in a mapping-based hardware implementation, the perturbations are contained in a look-up table and would result in a variable replacement in the mapping function. In a software implementation of the preferred embodiment, the control matrix M would be stored along with the software computing the chaotic dynamics so that when a perturbation is required, the information would be read from M.

A further improvement involves the use of microcontrols. Each time a trajectory of a chaotic system passes through a cross-section, the simulation is backed-up one time step, and the roles of time and space are reversed in the Runge-Kutta solver so that the trajectory can be integrated exactly onto the cross-section without any interpolation. Then, at each intersection where no control is applied, the trajectory is reset so that it starts at the central point of whatever bin it is in. This resetting process can be considered the imposition of microcontrols. It removes any accumulation of round-off error and minimizes the effects of sensitive dependence on initial conditions. It also has the effect of restricting the dynamics to a finite subset of the full chaotic attractor although the dynamics still visit the full phase

space. These restrictions can be relaxed by calculating r(x) and M to greater precision at the outset.

Another embodiment of a chaotic system utilizes an approximate one-dimensional Poincare map. The Poincare section has two branches, one on each lobe of the attractor. The partitioning of the surface and the use of microcontrols allow one to calculate easily a map that exhibits all of the symbolic dynamics of the full microcontrolled system. The evaluation of this map is much simpler and faster than integrating between intersections with the surface of section. To find the map, one can take the center point in each bin as an initial condition (since these are the points to which the micro controls "reset" trajectories), integrate forward in time until the next intersection with either branch of the surface of section, and note the branch and bin in which the trajectory landed. For a given set of integration parameters (time step, method, etc.) and for a given partition of the surface of section, the trajectory from the center of any bin to its next intersection with the surface will not vary. Therefore, the map mimics exactly the behavior of the microcontrolled system for the given integration method.

To implement this map, two more columns are placed in the instruction matrix M: one containing the row number in M that corresponds to the next intersection for all 2001 bins, and the other containing the next lobe under the map. Simulated data transmission and reception using this new matrix is essentially the same as transmission and reception using integration. For a given bin on the section and for a given message bit, the transmitter-encoder still uses the function r(x) to compare the symbolic dynamics N bits in the future. If the N-th bit in the future dynamics for that bin differs from the current message bit, r(x) is used to find the nearest bin that will produce the desired sequence. Then the map is used to find the location of the next intersection with the surface, and the process is repeated with the next message bit. The use of this

map eliminates time-consuming numerical integration, allowing for faster and more extensive processing.

The above map differs from a conventional Poincare map in a couple of aspects. First, while the Poincare section is two-dimensional, it is being approximated with a pair of lines extending from the unstable fixed points fitted with a least-squares method. Whenever a trajectory intersects the section, by only considering the distance from the corresponding fixed point, the point of intersection is essentially rotated about the fixed point onto the line before proceeding. Therefore the three-dimensional dynamical system is reduced to a one-dimensional map. Secondly, the point is reset to the center of its current bin to simulate the microcontrols. Theoretically, letting the maximum length of the intervals in the partition go to zero would make this second approximation unnecessary.

The use of a Poincare map allows a generalization of the system to any chaotic one-dimensional map. It is simply a matter of defining "lobes"-what section of the domain implies a switching of bits, recording the symbolic dynamics in r(x) and finding appropriate controls as before. For example, one could take the logistics map $x_n = ax_{n-1}(1-x_{n-1})$ and somewhat arbitrarily say that for any x_k x_{lobe}, where $0 < x_{lobe} < 1$, the current bit b_k will be $1 - b_{k-1}$: -otherwise, $b_k = b_{k-1}$. This gives the symbolic dynamics necessary to build a system, which can be improved in at least two ways. First, maps can be chosen that would have little to no grammar restriction, which would eliminate the need to adjust the bit stream to comply with the system's dynamics. Second, it is possible to fine-tune the maps to optimize the system statistically.

To eliminate the restriction that bits must at least come in pairs, it is necessary that the map allow trajectories to remain in the "switching" region for two or more iterations in a row. For example, one can use the second iterate of the logistics map, $x_n = a^2x_{n-1}(1 - x_{n-1})(1 - ax_{n-1}(1 - x_{n-1}))$, with $a = 3.99$. To preserve the

symmetry, it is logical to choose $x_{lobe} = 0.5$. All short N-bit words are possible in the natural evolution of this map, at least for $N < 4$, so there are no grammar restrictions with this system.

The chaotic system in the preferred embodiment described above had two lobes, labeled 0 and 1. Other chaotic systems can have more than two labels, and each lobe can be labeled 0 or 1 so that a bitstream is generated as each such lobe is visited by the trajectory of the system. Alternatively, each lobe can be assigned a symbol from any chosen symbol set so that a symbol sequence is generated by the trajectory of the system. The string of symbols can be converted into a bistream according to an appropriate algorithm.

In another embodiment, starting with the chaotic system in the preferred embodiment described above, rather than labeling the lobes of the chaotic system, one can label the control bins on the control surfaces. The bins can be labeled 0 or 1; or each bin can be assigned a symbol from any chosen symbol set. Then a bitstream is generated by the trajectory of the chaotic system, as described above. The trajectory of a chaotic system can also be used in other ways to generate a bitstream. For example, using the chaotic system in the preferred embodiment described above, one can track the intersections of the trajectory with the control surfaces and compare the i-th intersection with the (i+1)-th intersection and use a distance measure between the bins in which the intersections occurred to form an information string, which can be converted to a bitstream. As another example, one can apply a threshold to the amplitudes of the oscillations of the trajectory. Whenever an oscillation is above the threshold, a 1-bit is generated and whenever an oscillator is below the threshold a 0-bit is generated, resulting in a bitstream.

A chaotic system, such as those described above in the various embodiments, can be driven into synchrony with an identical chaotic system by the use of an initialization code. It is possible to send an initialization code, consisting of a sequence

of controls to each of the chaotic systems that will drive each of them onto the same periodic orbit. Once on the periodic orbit, an additional bit sent to it will cause it to leave the periodic orbit and generate a bitstream as described in detail above.

At a fundamental level, when microcontrols are used, there are only a finite number of orbits on a chaotic system, so periodicity of a chaotic system would eventually be guaranteed under a repeating sequence of controls. More importantly, a chaotic system can be driven onto a periodic orbit by sending it a repeating code. Different repeating codes lead to different periodic orbits. The periodic orbit reached is dependent only on the code segment that is repeated, and not on the initial state of the chaotic system (although the time to get on the periodic orbit can vary depending on the initial state). Consequently, it is possible to send an initialization code to two chaotic systems that drives them onto the same periodic orbit.

There are numerous control sequences that, when repeated, lead to a unique periodic orbit for all initial states, so that there is a one-to-one association between a sequence and the orbit. However, for some control sequences the orbits themselves change as the initial state of the chaotic system changes. Consequently, repeated control sequences can be divided into two classes, initializing codes and non-initializing codes. The length of each periodic orbit is an integer multiple of the length of the repeated control sequence. This is natural, since periodicity is attained only when both the current position on the cross-section *as well as* the current position in the control sequence is the same as at some previous time. To guarantee that the two chaotic systems are synchronized, it is sufficient that the period of the orbit is exactly the length of the smallest repeated segment of the initializing control sequence. Otherwise, it is possible that the two chaotic systems could be on the same periodic orbit, yet out of phase. Consequently, the chaotic systems would not be truly synchronized.

Chaotic systems can be implemented entirely in software. The chaotic systems in such an implementation are defined by a set of differential equations governing the

chaotic dynamics, e.g., the double scroll equations described above. An algorithm is used to simulate the evolution of the differential equations, e.g., the fourth order Runge-Kutta algorithm. In a second software implementation, mappings instead of differential equations can be used to define the chaotic systems. In this case, the chaotic systems are defined to take an input value and produce an output value.

Chaotic systems can also be implemented in hardware. The chaotic systems are still defined by a set of differential equations, but these equations are then used to develop an electrical circuit that will generate the same chaotic dynamics. The procedure for conversion of a differential equation into an equivalent circuit is well-known and can be accomplished with operational amplifiers and multipliers, as well as other devices known to one skilled in the art, configured with the proper feedbacks. The control information is stored in a memory device, and controls are applied by increasing voltage or inducing small current surges in the circuit. In a second hardware implementation, a mapping function is converted into a look-up table that can be stored on a digital memory chip, along with a table containing the control information. A message is encoded by using the look-up table to generate the chaotic dynamics.

A chaotic system can also be implemented in lasers. In this implementation, a set of differential equations is approximated using optical devices. Once the approximate system is developed, it defines the chaotic systems, and then control surfaces, partitions and microcontrols are defined for the chaotic dynamics realized by the laser system. The laser is driven into a chaotic mode of oscillation, and controls are developed using, e.g. the occasional proportional feedback ("OPF") technique. [E.R. Hunt Phys. Rev. Lett. 67, 1953 (1991)]. The control information is stored in a memory device that contains information defining the required controls for both the full controls and the microcontrols, as described above. The microcontrols are applied by

using, e.g., OPF controls to drive the chaotic dynamics toward the center of the partitions on the control surfaces.

Chaotic systems and initialization codes, preferred embodiments of both of which are described above, are used in a preferred embodiment of the present invention, as described in more detail below. Fig. 1 shows a remote digital key generating system 10 according to a preferred embodiment of the present invention. It comprises an encryptor 12 and a decryptor 14. It also comprises two identical chaotic systems, preferred embodiments of which are described above: a first chaotic system 16 and a second chaotic system 18. In operation, the encryptor 12 applies an initialization code 20, to the first chaotic system 16. The first chaotic system is allowed to generate an unpredictable first key bitstream 22 of desired length. The first key bitstream can be used as a digital key to encrypt a plain text message according to any of a number of encryption algorithms known to those skilled in the art. The encryptor 12 also sends the initialization code 20 to the decryptor 14 which applies the initialization code 20 to the second chaotic system 18. The second chaotic system 18 is then allowed to generate a second key bitstream 24, which second key bitstream 24 will be identical to the first key bitstream 22. The second key bitstream can be used as a digital key to decrypt text encrypted with the first key bitstream.

Fig. 2 is a flow chart of the remote digital key generation system of the present invention. In the first step 100, an initialization code, preferred embodiments of which are described above, is chosen. In step 102, the initialization code is applied to a first chaotic system to cause it to generate 104 an unpredictable first key bitstream of any desired length. The first key bitstream can be used as a digital key to encrypt a plain text message according to any of a number of encryption algorithms generally known in the art. The next step 106, which may occur before, at the same time, or after step 102, involves the application of the initialization code to a remote second chaotic system to cause it to generate 108 the second key bitstream, which is identical to the

first key bitstream, and can be used to decrypt any message encrypted with the first key bitstream.

These are many different key-based encryption algorithms known to those skilled in the art. They all involve the transmission of a key to the decrypting party or, as in the present invention, the transmission of a signal to the decrypting party allowing that party to generate the key. For example, public key encryption uses a public key-private key pair. The public key is used to encrypt a message, and the private key must be transmitted to, or generated remotely by, the decrypting party for decryption. In the case of the so-called knapsack algorithm, the decrypting party must receive, or generate, a super increasing sequence of numbers as a key for decryption. The present invention can be used to generate remotely a digital key for use in any key-based encryption algorithm. In addition, a key can be generated by combining a bitstream produced by the present invention, a bitstream generated according to the method of Short, *et al.* – Chaotic Communication, and an encrypting party's PIN number. The bitstreams can be combined to produce a key through a modulo addition of the binary numbers or any other operation on the bits.

The invention has been particularly shown and described above with reference to various preferred embodiments implementations and applications. The invention is not limited, however, to the embodiments, implementations or applications described above, and modification thereto may be made within the scope of the invention.